

1.Kill Processes by name

2.Kill a process based on the process name .

3.Kill a single process at a time with the given process ID

The screenshot shows a terminal window titled "shubh@kali: ~". The terminal is displaying the usage information for the "killall" command. The user has run the command "killall firefox" followed by the command "killall". The output shows the usage details for killall, including options like -e, -I, -g, -y, -o, -i, -l, -q, -r, -s, -u, -v, -V, -w, -n, -Z, and context REGEXP. The terminal prompt is visible at the bottom.

```
(shubh㉿kali)-[~]
$ pkill firefox

(shubh㉿kali)-[~]
$ killall
Usage: killall [OPTION] ... [--] NAME ...
      killall -l, --list
      killall -V, --version
      -e,--exact          require exact match for very long names
      -I,--ignore-case    case insensitive process name match
      -g,--process-group  kill process group instead of process
      -y,--younger-than   kill processes younger than TIME
      -o,--older-than     kill processes older than TIME
      -i,--interactive    ask for confirmation before killing
      -l,--list            list all known signal names
      -q,--quiet           don't print complaints
      -r,--regexp          interpret NAME as an extended regular expression
      -s,--signal SIGNAL   send this signal instead of SIGTERM
      -u,--user USER       kill only process(es) running as USER
      -v,--verbose          report if the signal was successfully sent
      -V,--version          display version information
      -w,--wait             wait for processes to die
      -n,--ns PID           match processes that belong to the same namespaces
                            as PID
      -Z,--context REGEXP  kill only process(es) having context
                            (must precede other arguments)

(shubh㉿kali)-[~]
$
```

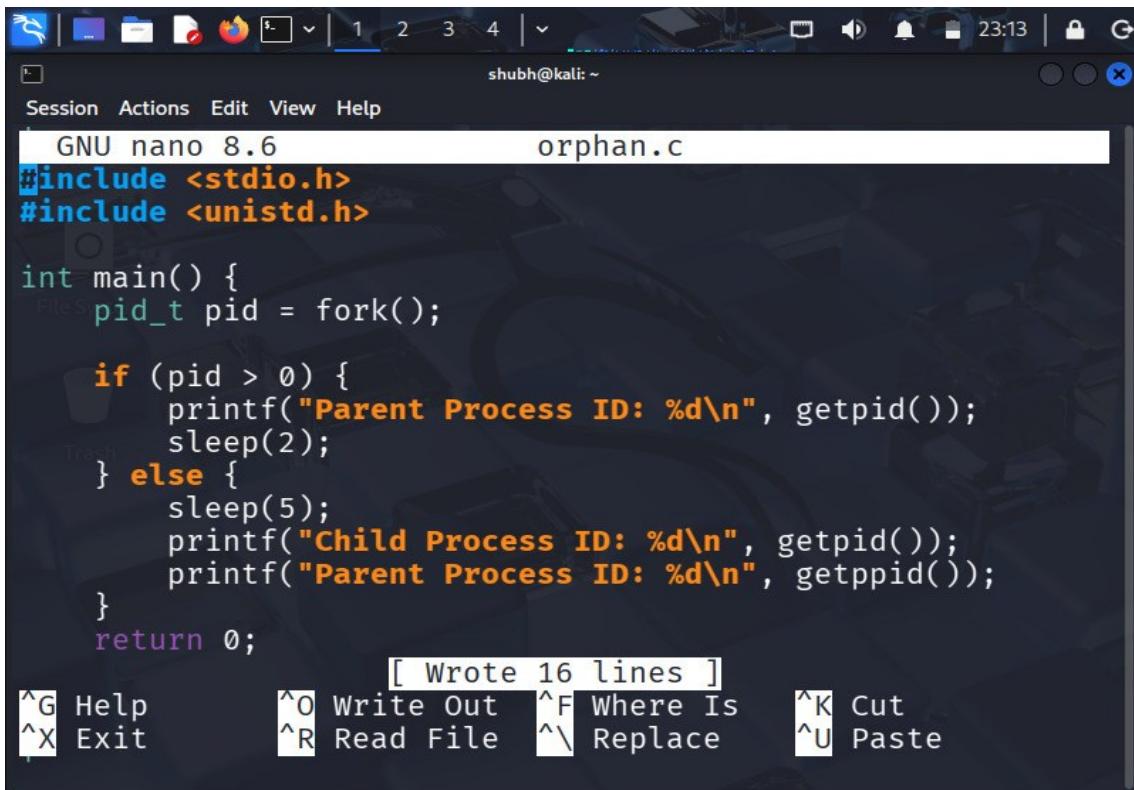
The screenshot shows a terminal window titled "shubh@kali: ~". The user has run the command "kill -9 2456", which results in an error message stating "kill: kill 2456 failed: no such process". The terminal prompt is visible at the bottom.

```
(shubh㉿kali)-[~]
$ kill -9 2456
kill: kill 2456 failed: no such process

(shubh㉿kali)-[~]
$
```

2. Write a program for process creation using C

1. Orphan Process 2. Zombie process



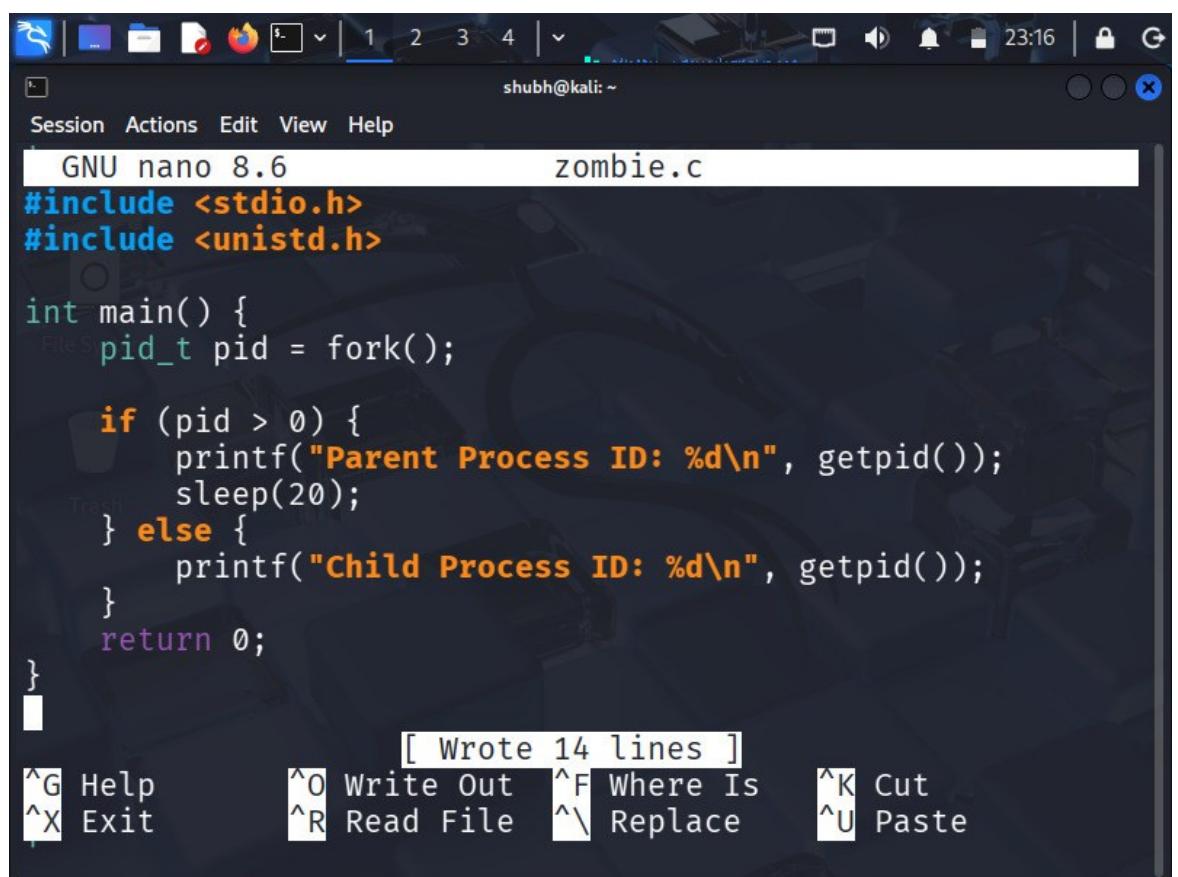
The screenshot shows a terminal window with a dark theme. At the top, there's a menu bar with "Session", "Actions", "Edit", "View", and "Help". The title bar says "shubh@kali: ~" and the file name is "orphan.c". The code in the editor is:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(2);
    } else {
        sleep(5);
        printf("Child Process ID: %d\n", getpid());
        printf("Parent Process ID: %d\n", getppid());
    }
    return 0;
}
```

At the bottom of the terminal, there are several keyboard shortcuts: ^G Help, ^X Exit, ^O Write Out, ^R Read File, ^F Where Is, ^\ Replace, ^K Cut, and ^U Paste. A message "[ Wrote 16 lines ]" is displayed at the bottom.



The screenshot shows a terminal window with a dark theme, similar to the one above. The title bar says "shubh@kali: ~" and the file name is "zombie.c". The code in the editor is:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(20);
    } else {
        printf("Child Process ID: %d\n", getpid());
    }
    return 0;
}
```

At the bottom of the terminal, there are several keyboard shortcuts: ^G Help, ^X Exit, ^O Write Out, ^R Read File, ^F Where Is, ^\ Replace, ^K Cut, and ^U Paste. A message "[ Wrote 14 lines ]" is displayed at the bottom.

### 3. Create the process using fork()system cell

1.Child Process creation 2.Parent Process creation

shubh@kali: ~

```
Session Actions Edit View Help
└$ gcc orphan.c -o orphan
cc1: fatal error: orphan.c: No such file or directory
compilation terminated.

└(shubh@kali)-[~]
└$ ls
Desktop Documents Downloads Music Pictures process.c

└(shubh@kali)-[~]
└$ gcc process.c -o orphan

└(shubh@kali)-[~]
└$ nano orphan.c

└(shubh@kali)-[~]
└$ gcc process.c -o orphan

└(shubh@kali)-[~]
└$
```

shubh@kali: ~

```
Session Actions Edit View Help
└$ gcc process.c -o orphan

└(shubh@kali)-[~]
└$ nano orphan.c

└(shubh@kali)-[~]
└$ gcc process.c -o orphan

└(shubh@kali)-[~]
└$ nano zombie.c

└(shubh@kali)-[~]
└$ gcc zombie.c -o zombie

└(shubh@kali)-[~]
└$ ./zombie
Parent Process ID: 21580
Child Process ID: 21581
```

```
shubh@kali: ~
Session Actions Edit View Help
└──(shubh㉿kali)-[~]
    $ nano zombie.c
└──(shubh㉿kali)-[~]
    $ gcc zombie.c -o zombie
File System
└──(shubh㉿kali)-[~]
    $ ./zombie
Parent Process ID: 21580
Child Process ID: 21581

└──(shubh㉿kali)-[~]
    $ ./orphan
Parent Process ID: 22125

└──(shubh㉿kali)-[~]
    $ Child Process ID: 22126
Parent Process ID: 1
```

```
shubh@kali: ~
Session Actions Edit View Help
nano fork.c
└──(shubh㉿kali)-[~]
    $ nano fork.c
└──(shubh㉿kali)-[~]
    $ gcc fork.c -o fork
└──(shubh㉿kali)-[~]
    $ ./fork
Parent Process
PID: 26532
Child PID: 26533

Child Process
PID: 26533
Parent PID: 1
└──(shubh㉿kali)-[~]
    $
```

The screenshot shows a Kali Linux desktop environment with a terminal window in the foreground. The terminal window is titled "fork.c" and displays a C program using the `fork()` system call to differentiate between parent and child processes. The code prints "Child Process\n", "PID: <pid>\n", and "Parent PID: <parent\_pid>\n". The terminal window also shows standard nano editor key bindings at the bottom.

```
GNU nano 8.6          fork.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child Process\n");
        printf("PID: %d\n", getpid());
        printf("Parent PID: %d\n", getppid());
    } else {
        printf("Parent Process\n");
        printf("PID: %d\n", getpid());
        printf("Child PID: %d\n", pid);
    }
}

^G Help      ^O Write Out   ^F Where Is   ^K Cut
^X Exit      ^R Read File   ^\ Replace    ^U Paste
```

